intro to machine learning with google cloud

brettkoonce.com/talks

june 20th, 2020

overview

- goal: how to get started with machine learning on google cloud
- background/theory
- ways to get started
- google cloud demos
- recap, q+a

machine learning

- intersection of data + statistics + compute

- linear regression, random forests, gradient boosted trees

- high-dimensional data

- pca/svd reduction, kernel methods, feature engineering

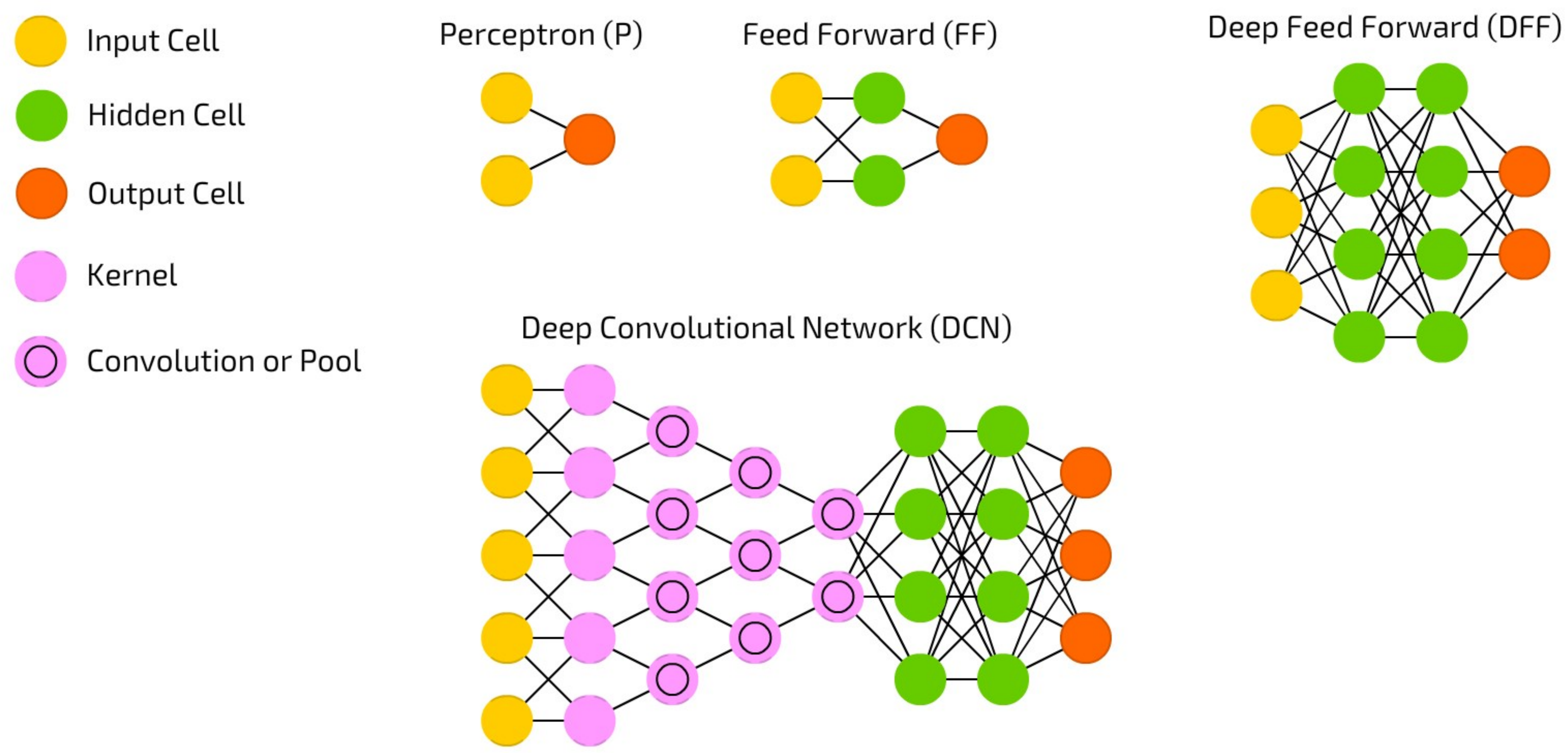
neural networks

- weight*x + bias --> a[X]+b
- activation functions
- can map to infinite data…
    - expensive to build/train!

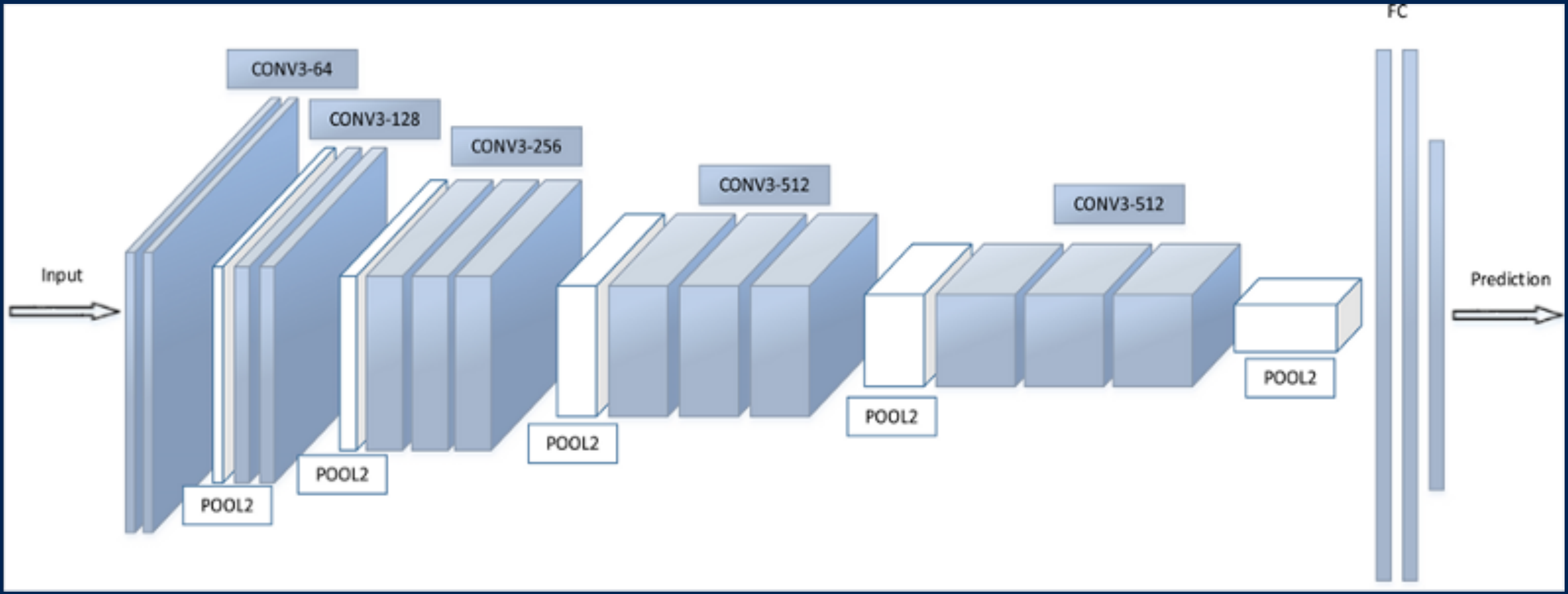
deep learning (2010's)

- commoditized compute (cpu/gpu)
- big data: # samples, types, resolution
- large scale software, commercial applications
- bitter lesson: simple >> complex

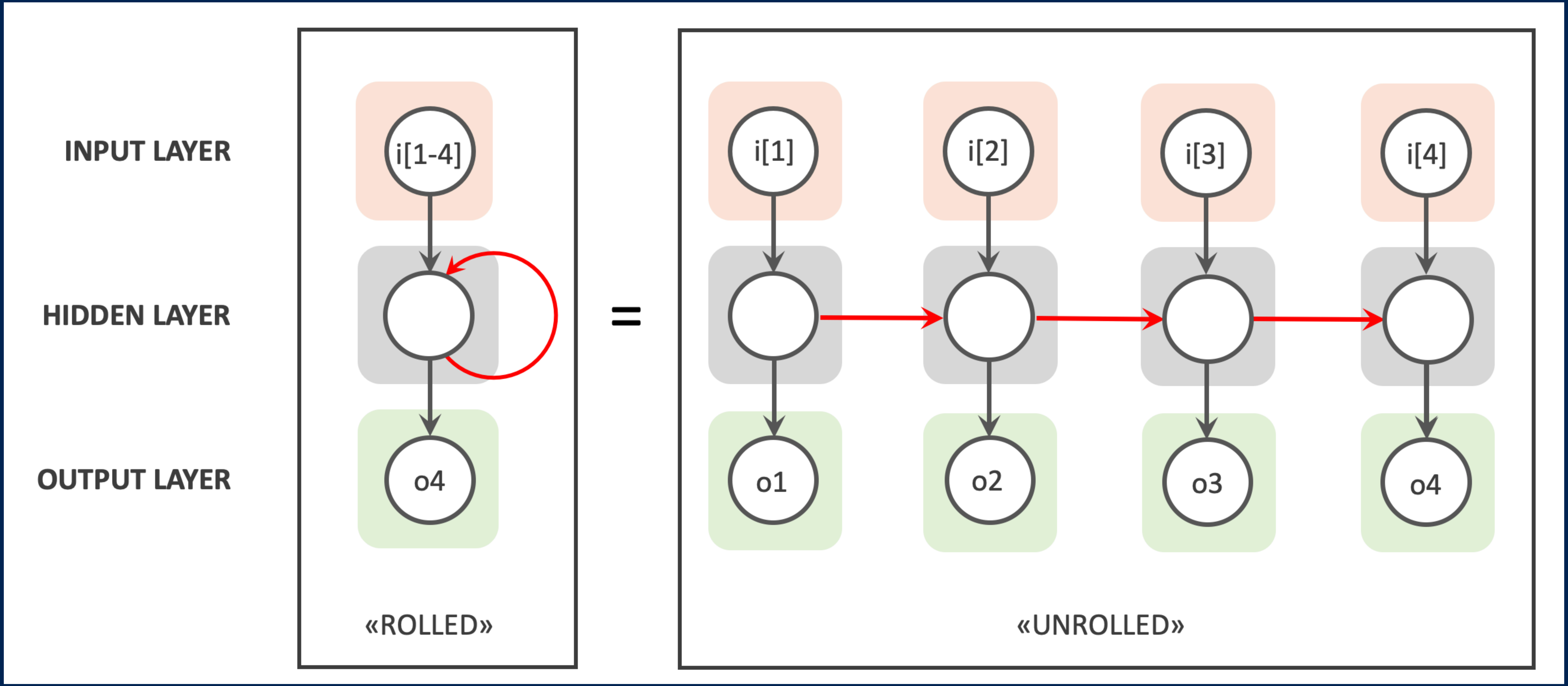
convolutional neural networks



vgg (2014)


Input
CONV3-64
POOL2
CONV3-128
POOL2
CONV3-256
POOL2
CONV3-512
POOL2
CONV3-512
POOL2
FC
Prediction

recurrent neural networks



seq2seq (2014)

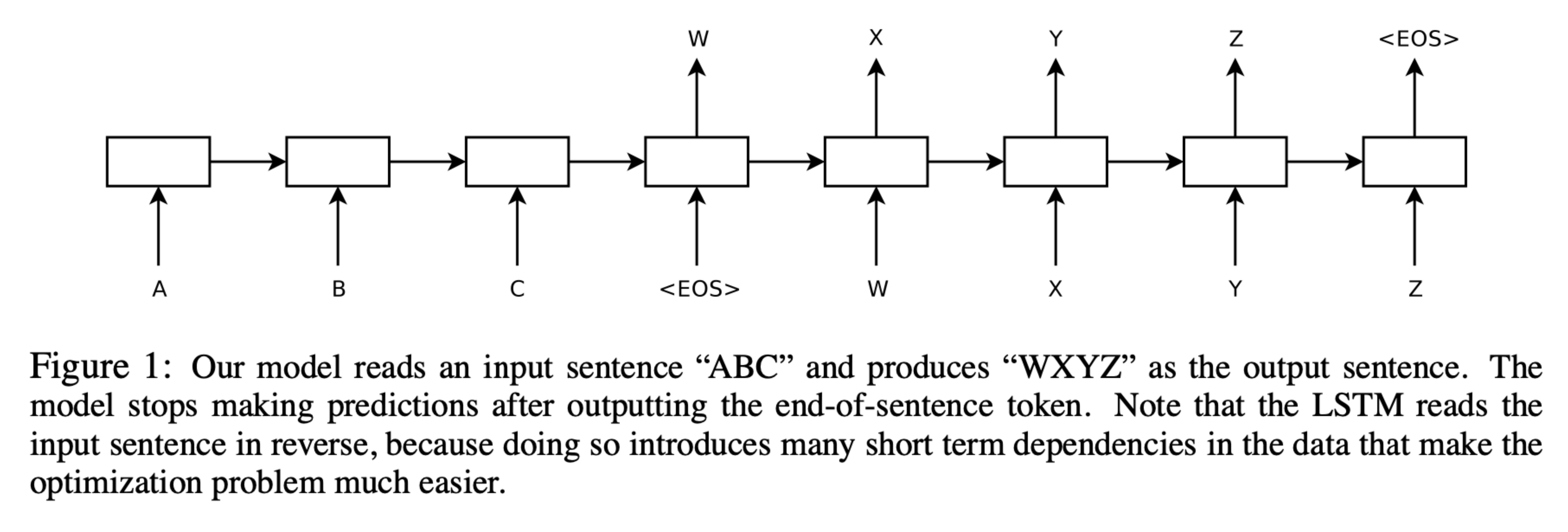


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

hybrid: detr (2020)

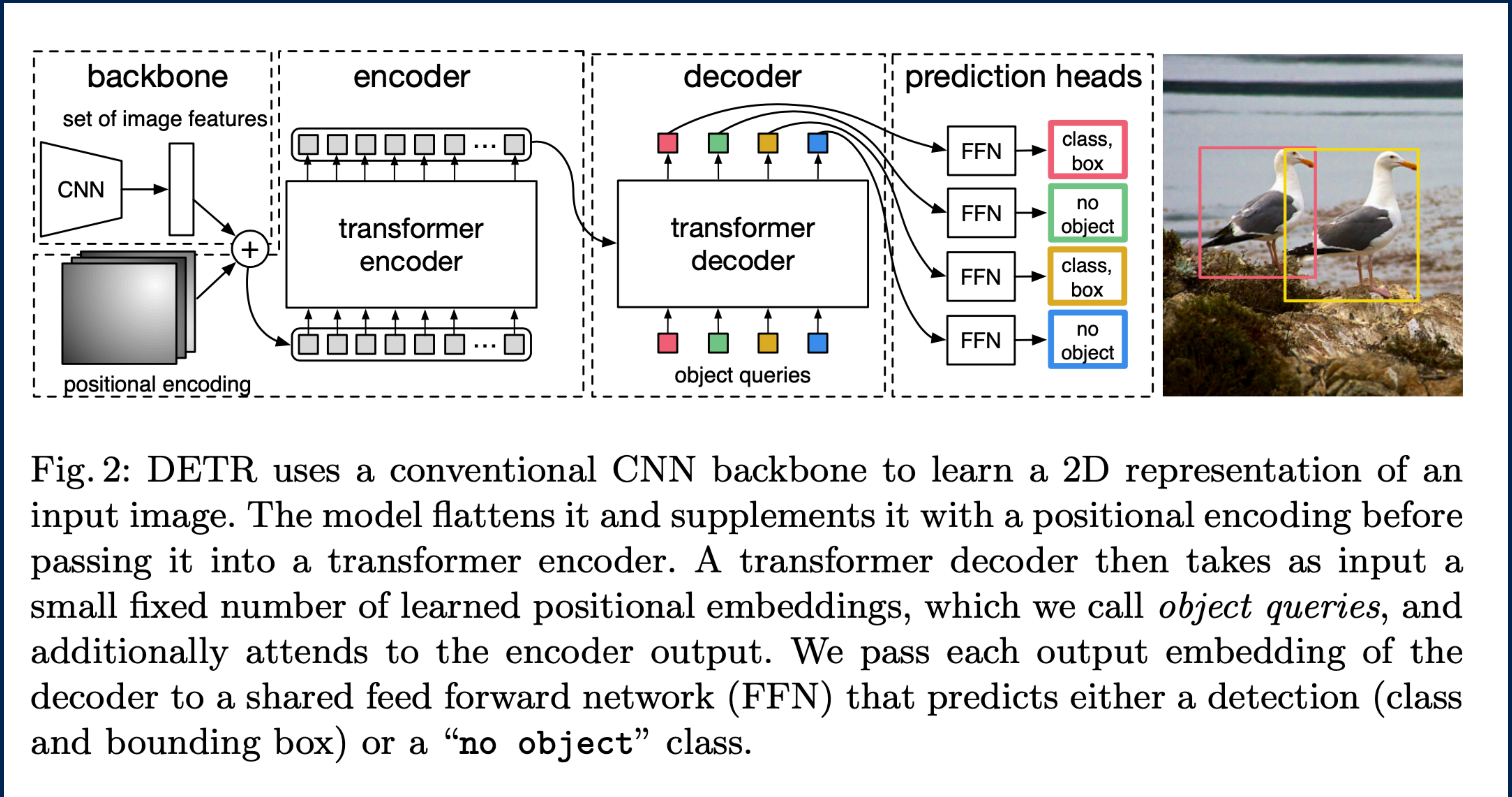
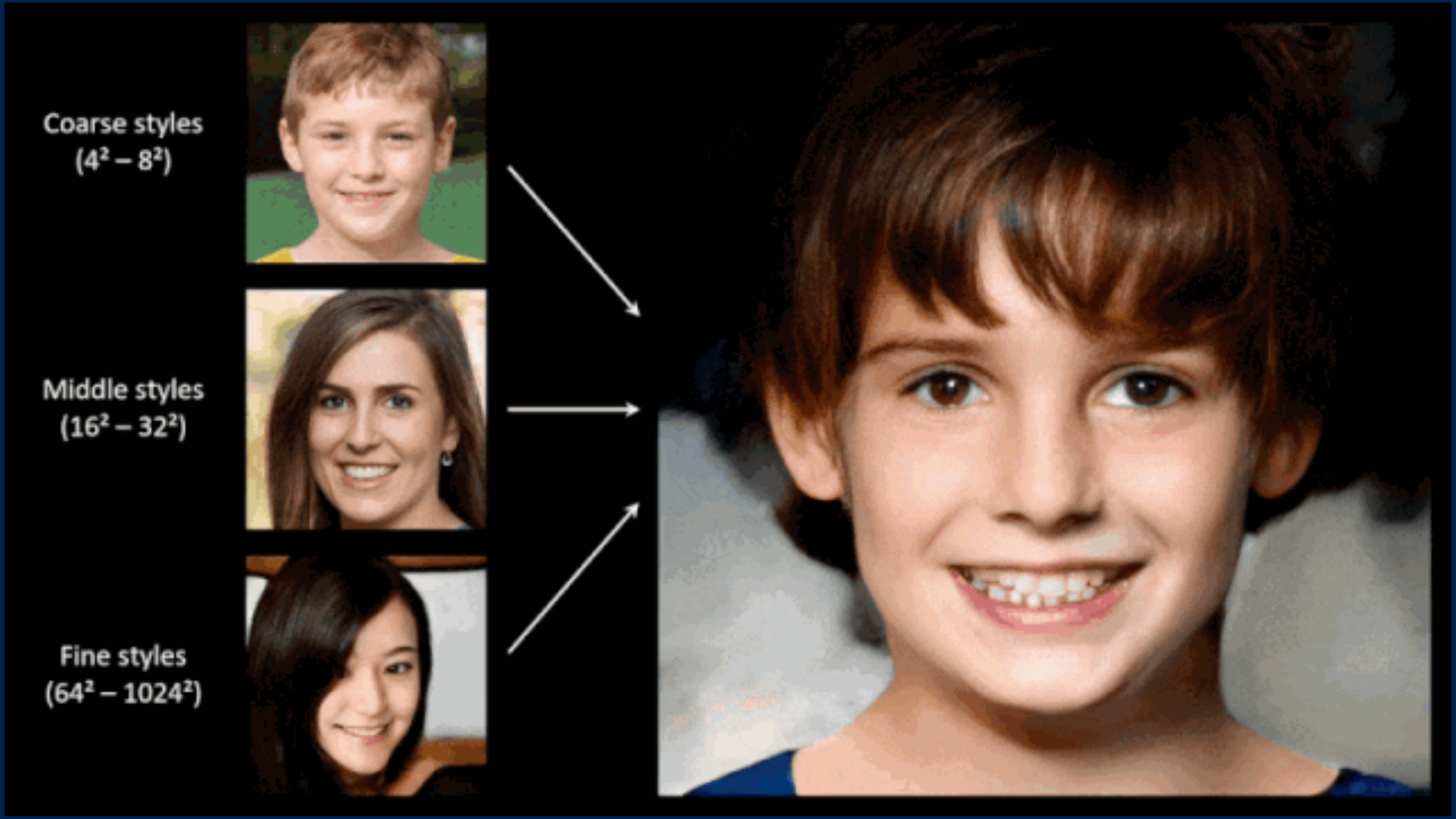
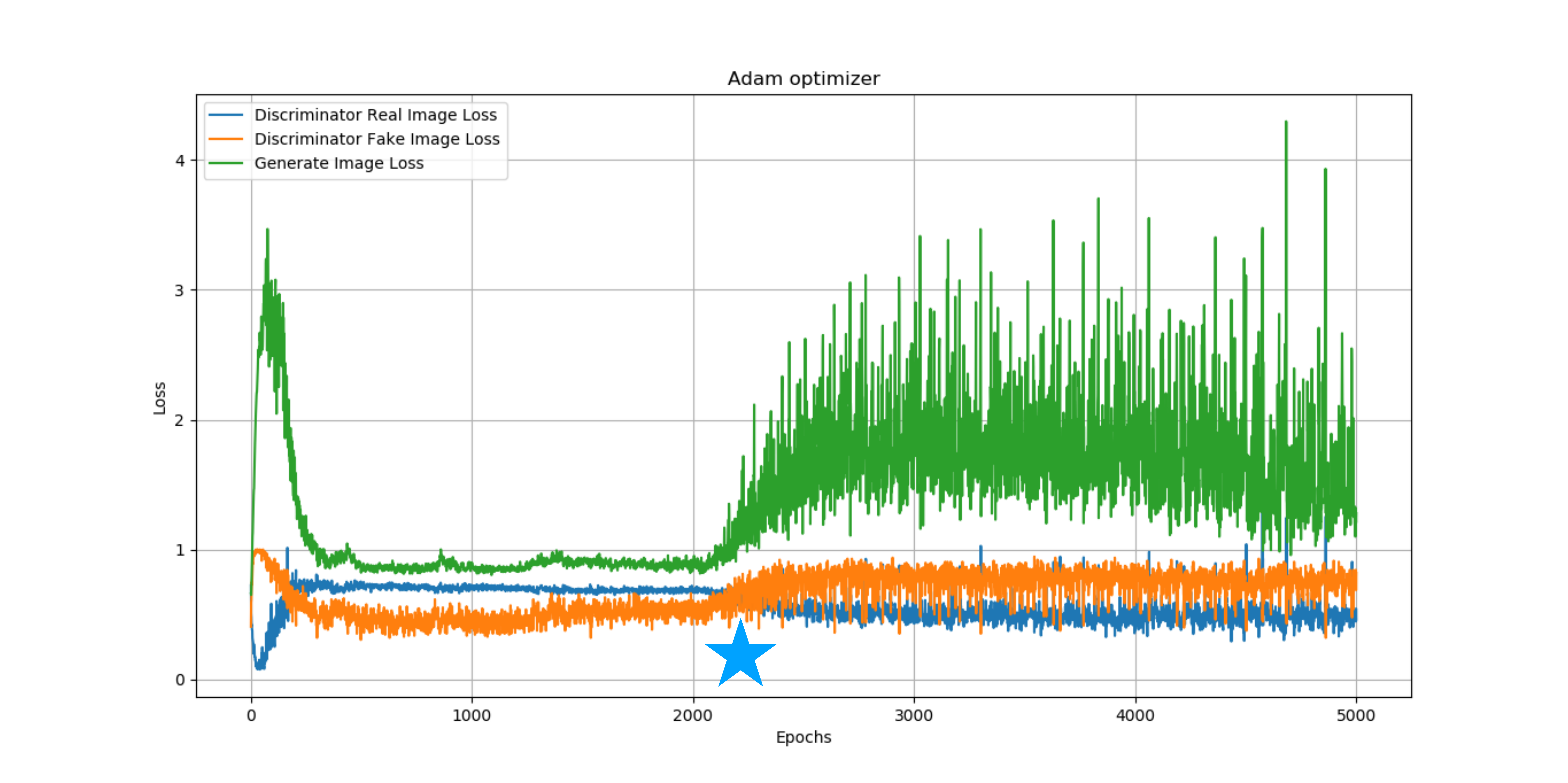


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

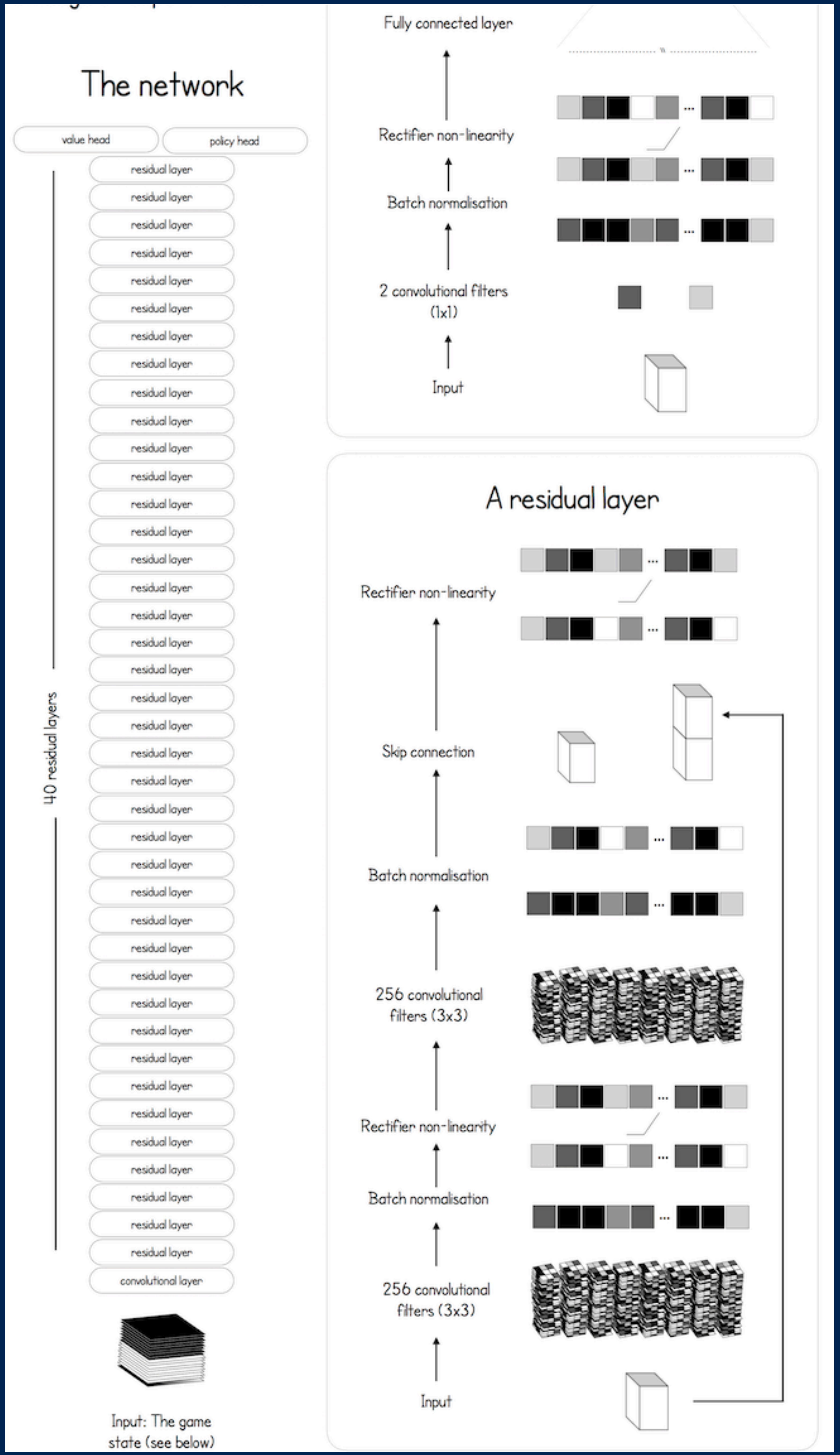
generative adversarial networks (2014, stylegan: 2018)

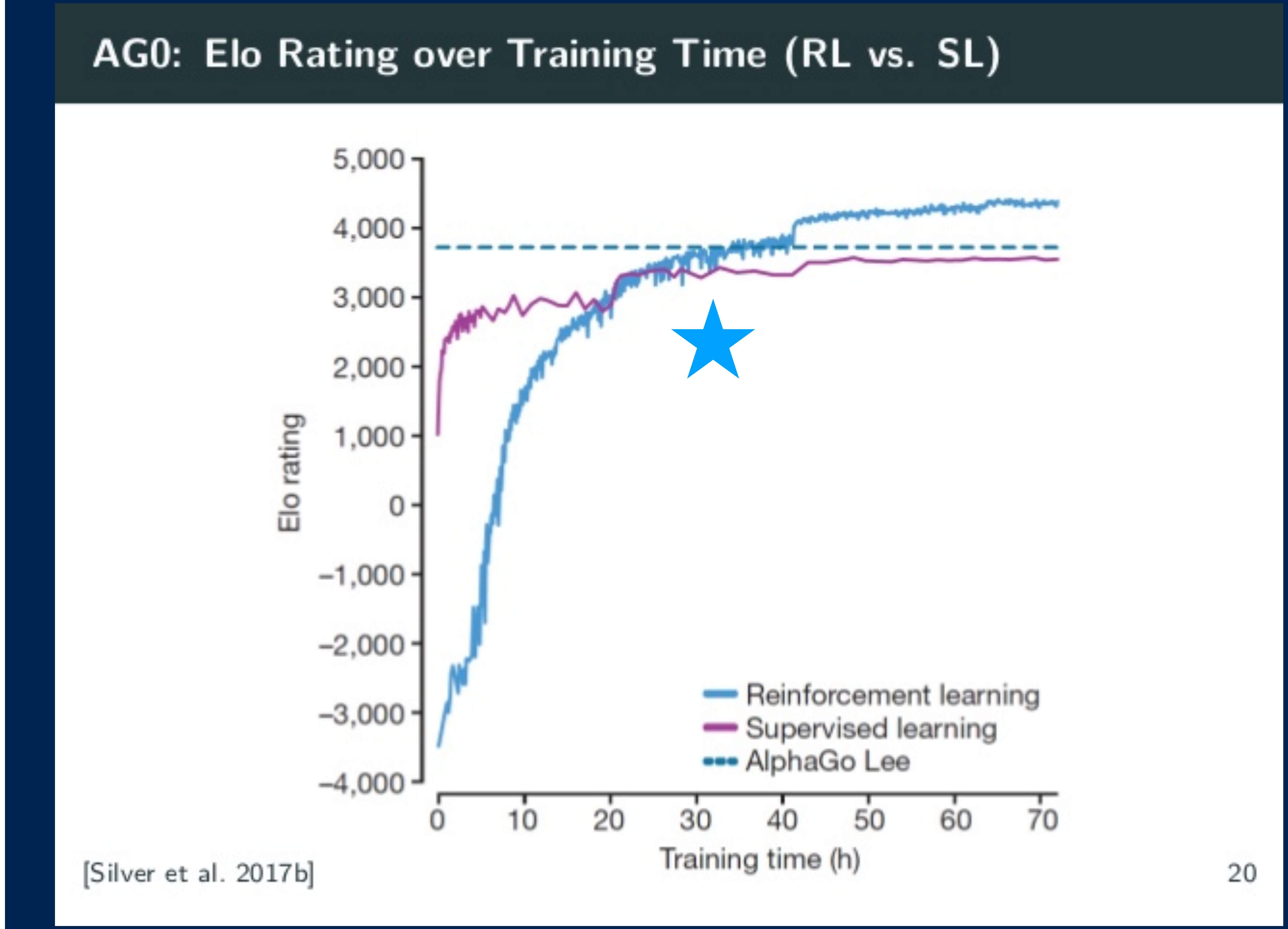


gan training loss



alphazero (2018)

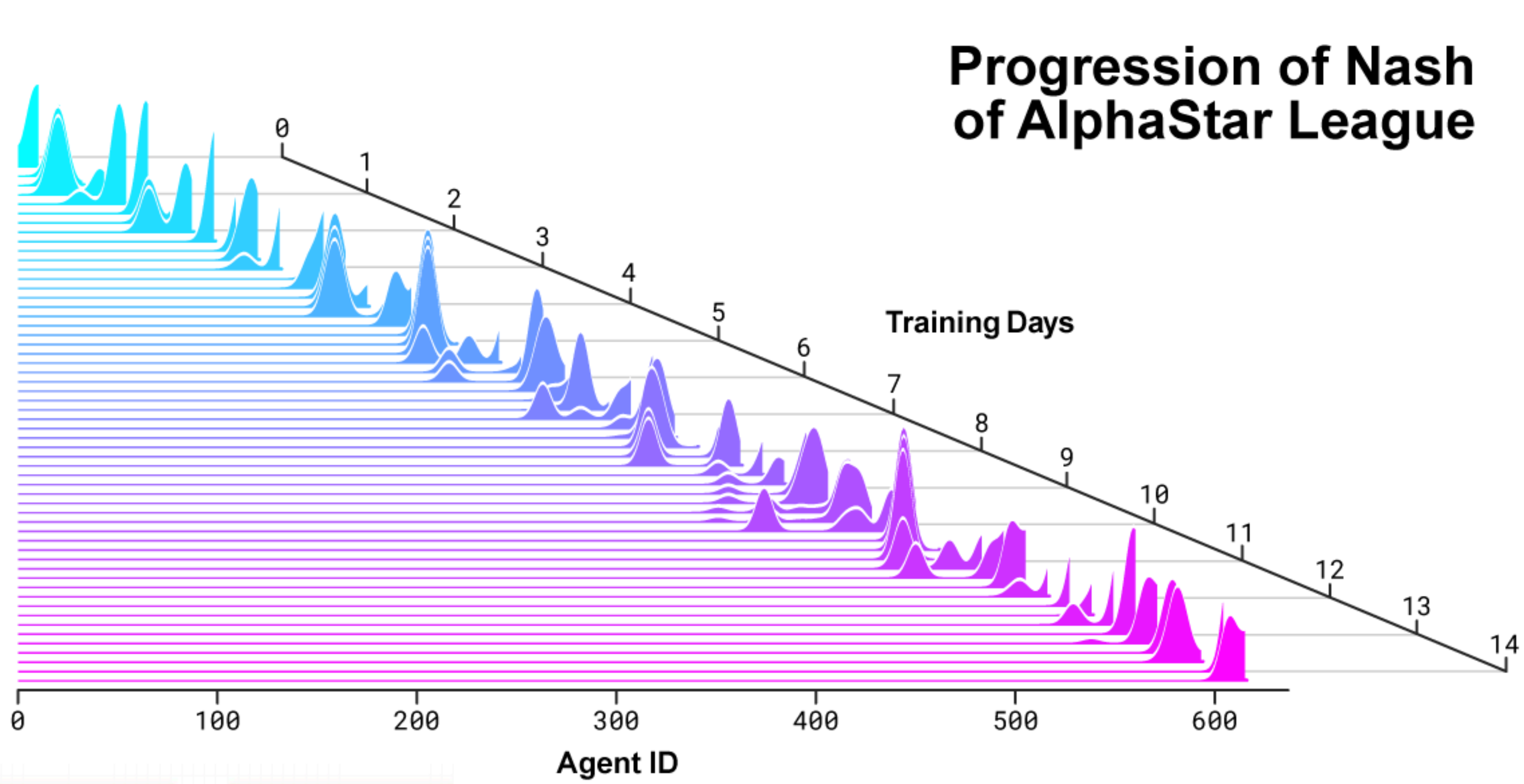


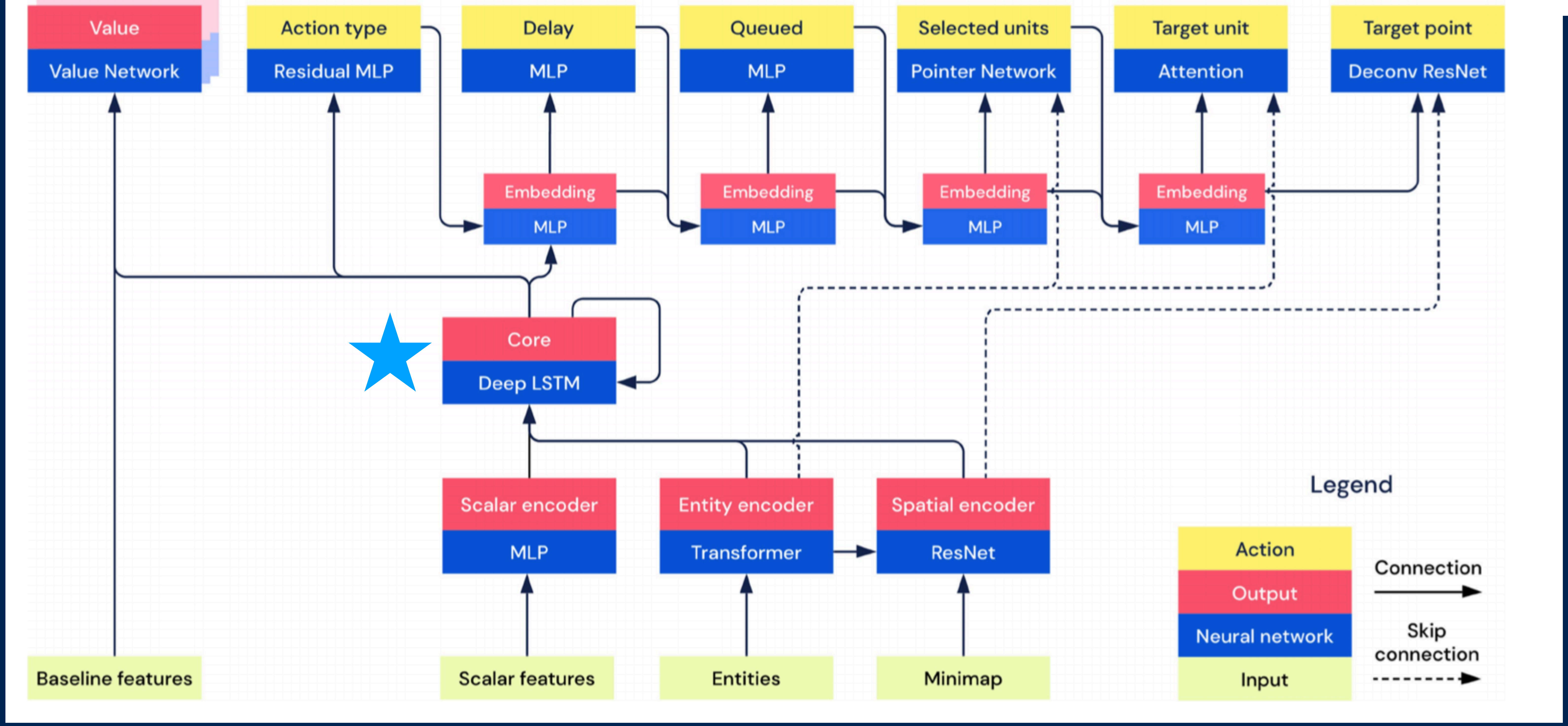


AG0: Elo Rating over Training Time (RL vs. SL)

[Silver et al. 2017b]

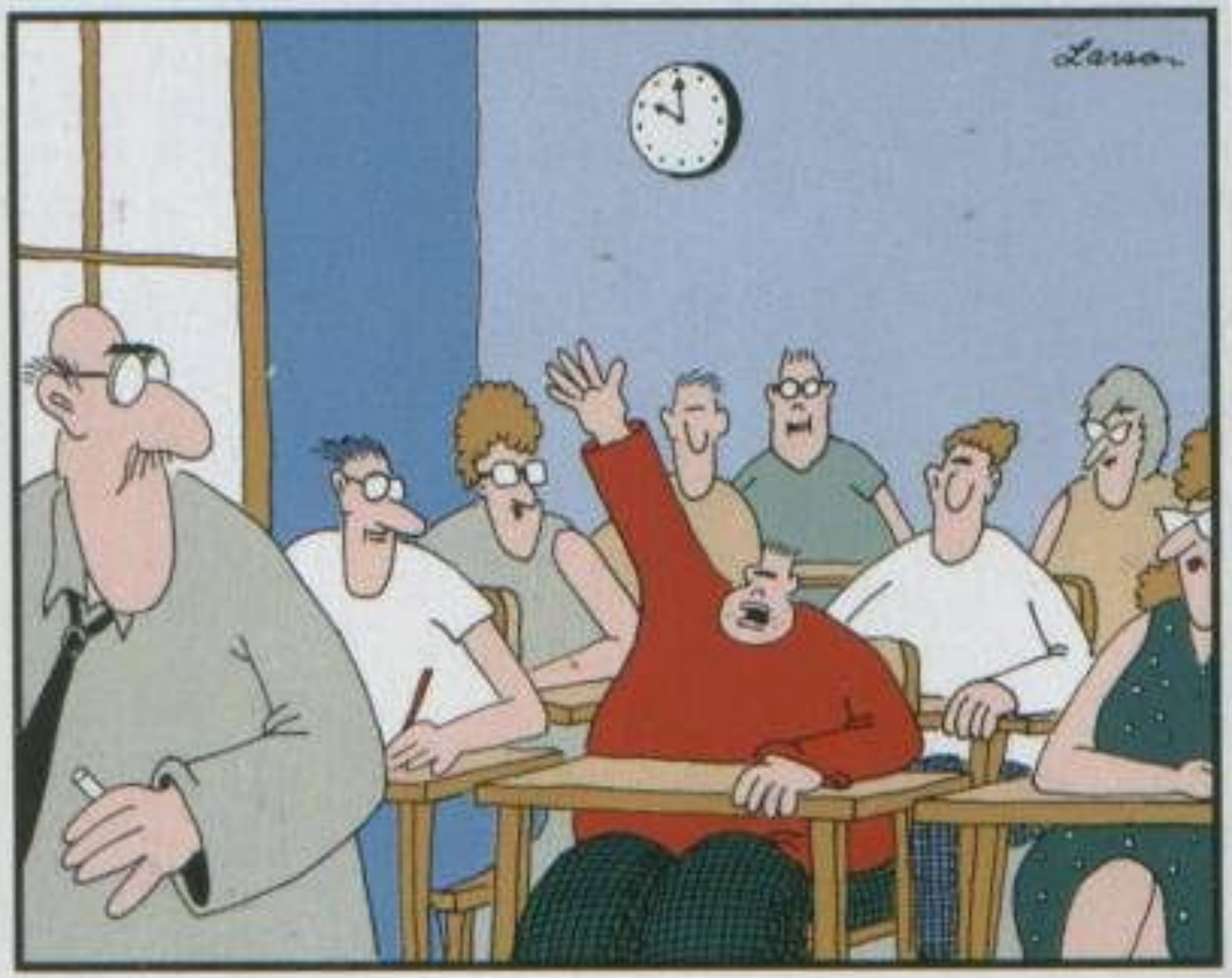
alphastar (2019)





artificial general intelligence (agi)

- key to ai: lots of compute power (??)
- are humans special?
- secrets of mother nature
- if ai is possible --> most important question of our time
- ten years ago <----> ten years ahead



"Mr. Osborne, may I be excused?
My brain is full."

don't panic!

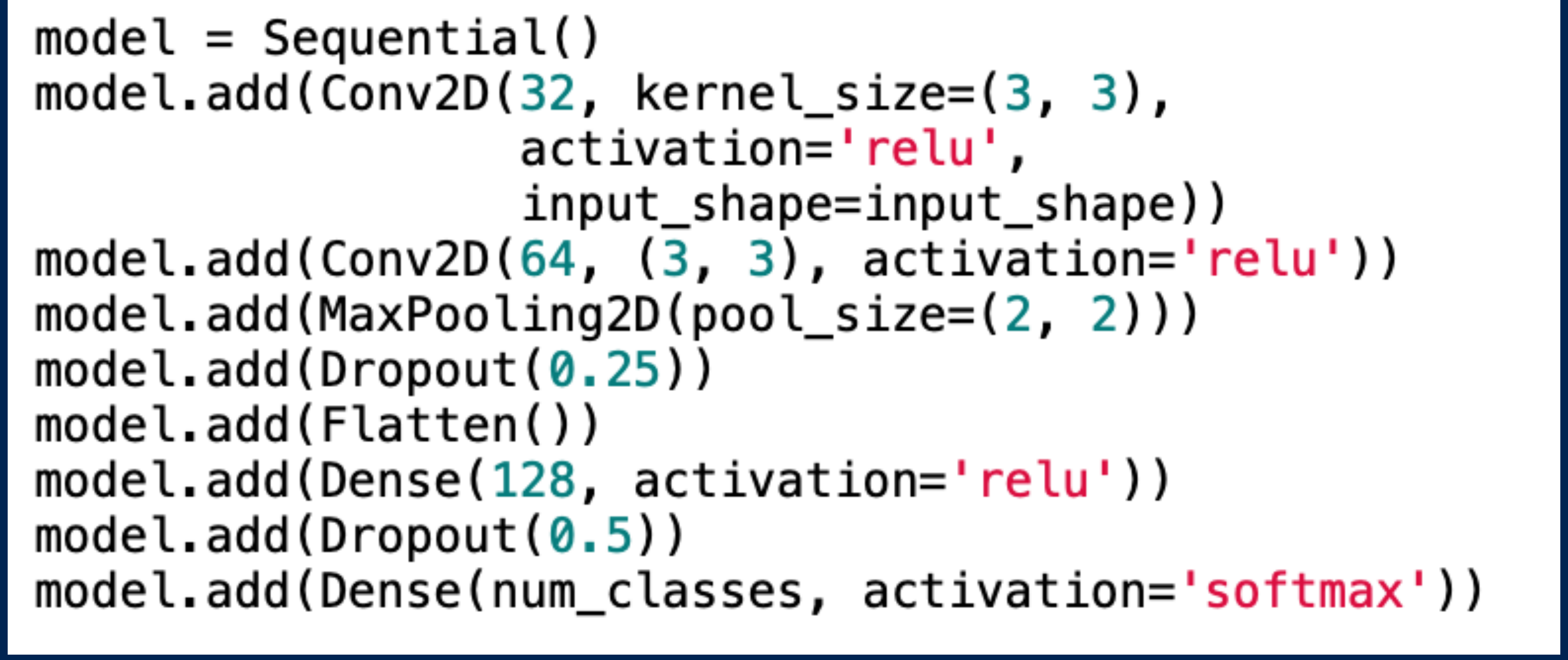
- anybody can do this!
- can learn basics for free
- focus on fundamentals, slowly add complexity
- follow herd, don't try to forge ahead

getting started

- pick a framework (tensorflow, pytorch)
- pick a tool (colab, google cloud, self-host)
- pick a teacher

tensorflow

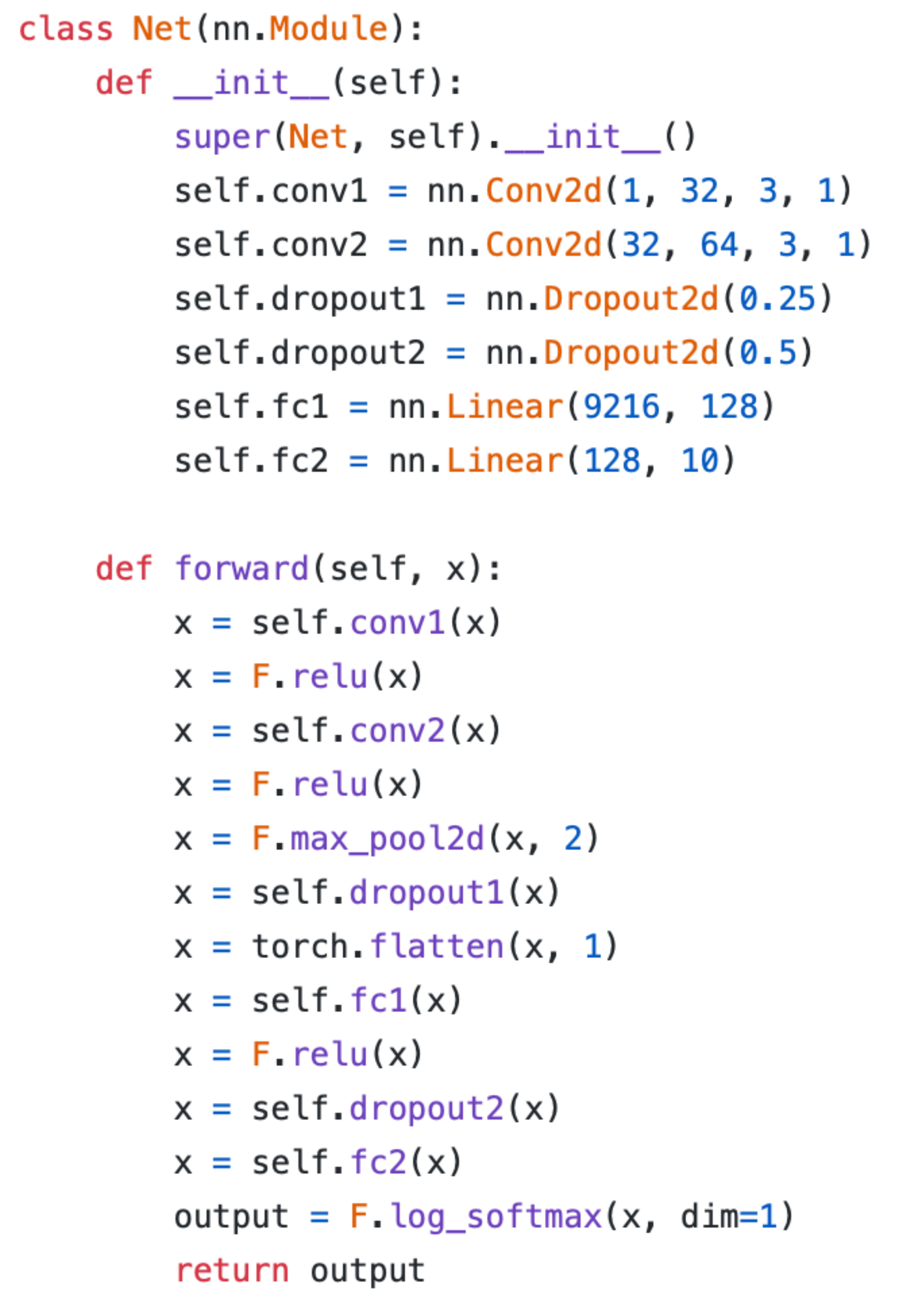
- tensorflow 1 vs 2 --> use 2.2 + python 3
- keras
- coursera + andrew ng
- google certificates



```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

pytorch

- python 3 w/ 1.5 and later
- jeremy howard + fast.ai



```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout2d(0.25)
        self.dropout2 = nn.Dropout2d(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output
```

frontier

- other frameworks
- jax: numpy --> xla bridge
- s4tf + xla: automatic differentiation, types
- convolutionalneuralnetworkswithswift.com
  - apress, 2020

demo time

- google colab (notebook) demo
- kubeflow/ai notebooks
- google cloud tools (rest api endpoints)
- deep learning ami
- custom vm

recap

- machine learning, deep learning
- neural network variants
- tools/ways to get going
- different cloud tools/approaches

thanks for coming!